Computer Aided Geometric Design 99 (2022) 102157

Contents lists available at ScienceDirect

Computer Aided Geometric Design

journal homepage: www.elsevier.com/locate/cagd

An unrefinement algorithm for planar THB-spline parameterizations

Teymur Heydarov [a,*], Annalisa Buffa [b], Bert Jüttler [c]

[a] Institute of Applied Geometry, Johannes Kepler University, Linz, Austria
[b] MNS, Institute of Mathematics, École Polytechnique Fédérale de Lausanne, Switzerland
[c] Johann Radon Institute for Computational an Applied Mathematics, Linz, Austria

ARTICLE INFO

Article history:
Available online 6 October 2022

Keywords:
Domain parameterization
Isogeometric analysis
THB-splines
Unrefinement

ABSTRACT

Unrefinement is a tool that allows to perform faster numerical simulations by controlling the level of precision in the specified area. We introduce an algorithm that creates a coarser geometry from an initial regular geometry, which is represented with respect to THB-splines, so that the coarser geometry approximates the initial geometry with a given level of the precision and is regular. The algorithm is based on the unrefinement of cells of the parameter domain, on which the geometry is defined, thus obtaining the new subdivision of the domain (the new subdomain hierarchy). The algorithm uses local linear projectors and optimization techniques in order to ensure the regularity and the approximation properties.

1. Introduction

Introduced by Hughes et al. (2005), isogeometric analysis (IGA) is a method for numerical simulation that integrates the finite element method (FEM) with the mathematical technology of computer-aided design. Compared to the standard FEM, it possesses the advantages of preserving the exact geometry representation, and it has been noted that it tends to need fewer degrees of freedom to achieve the required accuracy. The mathematical foundations of IGA have been investigated thoroughly (Beirão da Veiga et al., 2014) and the technology has been applied to various areas (see Hughes, 2008, and the references cited therein).

The geometric modeling of planar domains, surfaces or volumetric domains is based on the use of tensor-product splines, the refinement of which may lead to a substantial increase of the number of degrees of freedom. One of the methods of minimizing this effect is the use of adaptive spline refinement.

Adaptive refinement allows to refine the splines locally, based on the given criteria. T-splines (Sederberg et al., 2003), locally refined splines (LR-splines) (Dokken et al., 2013), polynomial T-splines over hierarchical T-meshes (Deng et al., 2008) and hierarchical T-splines (Evans et al., 2015) are some of the methods that have been described in the rich literature on this topic. We will concentrate on hierarchical B-spline refinement, which can be traced back to a paper by Forsey and Bartels (1988). While the classical version of the hierarchical B-splines do not possess the partition of unity property, this has been resolved by the introduction of the truncation mechanism (Giannelli et al., 2012). Adaptive THB-refinement has been used in industrial applications by Kiss et al. (2014) and Bracco et al. (2018).

* Corresponding author.
E-mail addresses: teymur.heydarov@jku.at (T. Heydarov), annalisa.buffa@epfl.ch (A. Buffa), bert.juettler@jku.at (B. Jüttler).

https://doi.org/10.1016/j.cagd.2022.102157

In addition to adaptive refinement, it is sometimes also useful (e.g., to maintain the efficiency of numerical simulations) to perform the inverse operation, which is called unrefinement or coarsening (Scott et al., 2014). The aim of this procedure is to make geometries coarser – thereby reducing the number of degrees of freedom, which will lead to faster simulations – without sacrificing too much accuracy. For instance, the representation of the geometry may be excessively detailed in places that are not relevant for the numerical simulation, or for one of them if different aspects are to be studied. Clearly, this requires to relax somewhat the isogeometric principle of "exact" geometry representation, by allowing to omit irrelevant geometric details.

Unrefinement of THB-splines was studied by Garau and Vázquez (2018) and Hennig et al. (2018). It has been applied to additive manufacturing (Carraturo et al., 2019), phase-field models (Lorenzo et al., 2017), and topology optimization (Xie et al., 2021). The topic of defeaturing, which is the process of simplifying less relevant parts of the geometry, is particularly challenging but also highly useful (Buffa et al., 2020, 2022).

In this paper, we introduce an algorithm that creates a coarser geometry from an initial regular geometry so that the coarser geometry approximates the initial geometry with a given level of the precision. The algorithm is based on the unrefinement of the cells of the parameter domain, on which the geometry is defined, thus obtaining the new subdivision of the domain (the new subdomain hierarchy). The algorithm uses local linear projectors and optimization techniques in order to ensure the regularity and approximation properties. The obtained parameterization is to be used in further simulations.

The remainder of the paper is organized as follows. After providing some preliminaries on the subdivision of the computational domain and THB-splines in Section 2, we formulate the problem in the third section. The algorithm is introduced in Section 4 and a brief explanation of its steps is given. Section 5 provides further details, in particular concerning the local fitting and the optimization methods. Finally, experimental results are provided in Section 6 and used to analyze the performance of the algorithm.

2. Preliminaries

This section provides basic concepts about the hierarchical and THB-splines and some of their properties that will be used in the following Sections.

For a given number $\ell \in \{1, \dots, N\}$, which we call the *level*, we consider the subdivision of $\mathbb{R}^2$ into *cells* $c \in C^\ell$ of level ℓ, which are the elements of the set

$$C^\ell = 2^{1-\ell}([0,1]^2 + \mathbb{Z}^2) \, .$$

In addition, we consider the subset $S^\ell \subset C^\ell$ of the selected cells and use it to define the domains Ω^ℓ:

$$\Omega^\ell = \bigcup_{c \in S^\ell} c \, .$$

We will assume that the selection of the cells ensures that the resulting domains are nested,

$$\Omega^1 \supseteq \Omega^2 \supseteq \dots \supseteq \Omega^\ell \supseteq \dots \supseteq \Omega^N \supseteq \Omega^{N+1} \, . \tag{1}$$

Here we define $\Omega^{N+1} = \emptyset$ and we moreover require that $\Omega^1 = [0,1]^2$, which is the *domain* of the spline functions that we consider. The finite sequences of subdomains

$$\mathbf{\Omega} = \left(\Omega^\ell\right)_{\ell=1,\dots,N+1}$$

will be called the *subdomain hierarchies*. In addition to the cells, we consider tensor-product spline spaces

$$V^\ell = W^\ell \otimes W^\ell \, ,$$

where W^ℓ is the space of univariate splines with the bi-infinite knot sequence $2^{-\ell}\mathbb{Z}$. The latter space is spanned by B-splines, thus V^ℓ is spanned by a basis B^ℓ of tensor-product (TP) B-splines.

The construction of hierarchical B-splines uses a support-based selection procedure, where we use the support with respect to the domain $[0,1]^2$,

$$\operatorname{supp} f = \{\, x \in [0,1]^2 \, : \, f(x) \neq 0 \,\} \, . \tag{2}$$

A (TP)B-spline $\beta^\ell \in B^\ell$ is said to be *active* if its support lies entirely in the domain Ω^ℓ but not in the next finer one,

$$\beta^\ell \in B^\ell \text{ is active} \quad \Leftrightarrow \quad \operatorname{supp} \beta^\ell \subseteq \Omega^\ell \wedge \operatorname{supp} \beta^\ell \not\subseteq \Omega^{\ell+1} \, .$$

The active functions of each level ℓ form the set

$$H^\ell = \{\beta^\ell \in B^\ell \mid \beta^\ell \text{ is active}\} \, ,$$

and the union of these sets H^ℓ is the *hierarchical B-spline basis*

$$H = \left\{ \beta^\ell \mid \beta^\ell \in H^\ell, \quad \ell = 1, \dots, N \right\}$$

that spans the *hierarchical spline space* $\mathcal{H} = \operatorname{span} H$. In addition to (1), we will assume that the choice of the selected cells ensures

$$\Omega^\ell = \bigcup_{k=\ell}^{N} \bigcup_{\beta^k \in H^k} \operatorname{supp} \beta^k \,, \ell = 1, \dots, N \,,$$

i.e., the supports of the selected functions of level ℓ or higher cover the whole of Ω^ℓ.

Instead of the hierarchical B-spline basis, we will also consider the truncated version thereof, which is denoted by T^ℓ, and its elements by τ^ℓ. Every hierarchical B-spline has an associated truncated hierarchical (TH) B-spline, which is generated by subtracting hierarchical B-splines of higher levels. This construction, which is called *truncation*, ensures that the resulting system of THB-splines forms a non-negative partition of unity, see Giannelli et al. (2012) for details.

Theorem 1. *(Vuong et al., 2011) Consider the two sequence of nested domains*

$$\Omega^{N-1} \subseteq \Omega^{N-2} \subseteq \ldots \subseteq \Omega^0$$

and

$$\hat{\Omega}^{N-1} \subseteq \hat{\Omega}^{N-2} \subseteq \ldots \subseteq \hat{\Omega}^0$$

together with the corresponding hierarchical bases K^ℓ and $\hat{K}^\ell$ for $\ell = 0, \dots, N-1$. If $\Omega^\ell \subseteq \hat{\Omega}^\ell$ for $\ell = 0, \dots, N-1$ then

$$\operatorname{span} K^\ell \subseteq \operatorname{span} \hat{K}^\ell \,.$$

The set of all the hierarchical spline spaces is partially ordered with respect to inclusion. In addition, the inclusion defines a partial ordering on the set of all the subdomain hierarchies also,

$$\mathbf{\Omega} \subseteq \hat{\mathbf{\Omega}} \quad \text{if} \quad \Omega^\ell \subseteq \hat{\Omega}^\ell \,, \quad \forall \ell = 1, \dots, N \,.$$

(Note that $\Omega^1 = \hat{\Omega}^1 = [0,1]^2$ and $\Omega^{N+1} = \hat{\Omega}^{N+1} = \emptyset$). Theorem 1 ensures that these two orderings are equivalent. Consequently, we obtain larger (or smaller) spline spaces by enlarging (or shrinking) the subdomains.

3. The problem

Let $\mathbf{p}_0(u, v)$ be a parameterization of the domain of interest (e.g., of the computational domain in isogeometric analysis) with parameters $(u, v) \in [0,1]^2$. The aim of this paper is to establish a geometric unrefinement algorithm that modifies the given parameterization $\mathbf{p}_0$ and creates a new parameterization $\mathbf{p}$

$$\mathbf{p}(u, v) = \sum_{i=1}^{N} \sum_{\tau^\ell \in T^\ell} \tau^\ell(u, v) \mathbf{d}_{\tau^\ell} \,, \tag{3}$$

with control points $\mathbf{d}_i \in \mathbb{R}^2$. We will assume that $\mathbf{p}_0$ is represented with respect to truncated hierarchical (TH)B-splines, which are defined by the initial subdomain hierarchy $\mathbf{\Omega}_0$. The basis functions of the new parameterization (3) are chosen as THB-splines with respect to a *coarser* hierarchy $\mathbf{\Omega}$.

The geometric unrefinement algorithm has the following main objective:

(MO) The numbers of degrees of freedom, i.e. the dimension of the spline space $\mathcal{H}$ defined by the subdomain hierarchy $\mathbf{\Omega}$, should be as small as possible. More precisely, we want to obtain a parameterization that has fewer degrees of freedom than the initial one. The less, the better, though we do not aim at treating this as a minimization problem.

In addition it has to satisfy the following three conditions:

(C1) The new parameterization (3) should be regular. In particular, no self-intersections are allowed to be present. This objective can be expressed as

$$\det(J(\mathbf{p}(u, v))) > 0 \quad \forall\, (u, v) \in [0,1]^2 \,.$$

(C2) The new parameterization (3) should approximate the initial parameterization. More precisely, the *global* approximation error should satisfy the inequality

$$\left\| \|\mathbf{p}_0(u,v) - \mathbf{p}(u,v)\|_E \right\|_{\infty,[0,1]^2} < \varepsilon_{\text{global}}$$

for some user-defined tolerance $\varepsilon_{\text{global}}$, where $\|.\|_E$ denotes the Euclidean norm.

(C3) The new parameterization (3) should provide a highly accurate approximation of the initial parameterization on some subset $F \subseteq \partial[0,1]^2$ of the boundary of the parameter domain, which we call the *feature*. More precisely, the approximation error is expected to satisfy the inequality

$$\left\| \|\mathbf{p}_0(u,v) - \mathbf{p}(u,v)\|_E \right\|_{\infty,F} < \varepsilon_{\text{feature}}$$

for some user-defined tolerance $\varepsilon_{\text{feature}}$, which is typically smaller than $\varepsilon_{\text{global}}$. For simulations we need to maintain a highly precise approximation of the representation of the boundary, or at least of the simulation–relevant part of the boundary.

This problem formulation includes two limit cases:

- If we choose $\varepsilon_{\text{global}} = \infty$, then we do not impose any constraint on the global approximation error.
- If we choose $F = \partial[0,1]^2$, then we consider the error on the entire boundary of the computational domain $\mathbf{p}_0([0,1]^2)$. This is the most relevant situation in applications.

4. The algorithm

We introduce the unrefinement algorithm and describe its different steps in detail. The initial parameterization is represented with respect to THB-splines defined by a given sequence of subdomains

$$\mathbf{\Omega}_0 = \left(\Omega_0^\ell\right)_{\ell=1,\dots,N+1},$$

which span the hierarchical spline space $\mathcal{H} = \operatorname{span} H$. The unrefinement is performed with the help of Algorithm 1, which proceeds as follows.

Algorithm 1: Rebuilding the hierarchical structure.

Input: The parameterization $\mathbf{p}_0$, the initial hierarchy $\mathbf{\Omega}_0$, the set of the eligible cells $\mathcal{E}^\ell$, tolerances $\varepsilon = (\varepsilon_{\text{global}}, \varepsilon_{\text{feature}})$, the feature F
Output: the coarsened hierarchy $\mathbf{\Omega}$, the modified parameterization $\mathbf{p} \in H[\mathbf{\Omega}]$

```
1  Initialize Ω̂, p̂, success;
2  success ← true, Ω ← Ω_0, p ← p_0;
3  for ℓ = N − 1 downto 1 do
4      Initialize E^ℓ as in Eq. (4);
5      while E^ℓ ≠ ∅ ∧ success do
6          success ← false, Ê^ℓ ← E^ℓ;
7          forall c ∈ Ê^ℓ do
8              Ê^ℓ ← Ê^ℓ \ {c};
9              Ω̂ ← modify Ω by subtracting c from Ω^{ℓ+1};
10             for i ← 1 to 3 do
11                 p̂ ← Projection[i](Ω̂, p_0);
12                 if valid(p̂, p_0, ε, F) then
13                     success ← true;
14                     p ← p̂, Ω ← Ω̂, E^ℓ ← Ê^ℓ;
15                     break;                                   // for i
```

- The algorithm takes the given parameterization $\mathbf{p}_0$, its subdomain hierarchy $\mathbf{\Omega}_0$, the feature F and the tolerances $\varepsilon = (\varepsilon_{\text{global}}, \varepsilon_{\text{feature}})$ as an input.
- First we initialize on line 1 the temporary subdomain hierarchy $\hat{\mathbf{\Omega}}$, the temporary parameterization $\hat{\mathbf{p}}$ and the auxiliary variable *success*, which is used to control the flow of the algorithm. Initially, *success* is set to be *true*. The modified parameterization and the coarser subdomain hierarchy are initialized to be the same as the given ones, see line 2.
- The outer **for** loop visits all the levels, starting from the finest one.

- Among the selected cells of the level ℓ we identify the ones that are eligible for unrefinement to level ℓ and collect them in the set

$$\mathcal{E}^\ell = \{c \in \mathcal{S}^\ell \mid c \subseteq \Omega^{\ell+1} \ \wedge \ c \cap \Omega^{\ell+2} = \emptyset\}, \quad \ell = 1, \dots, N-1 \,. \tag{4}$$

  These cells belong to $\Omega^{\ell+1}$ and can therefore be unrefined to level ℓ. We only consider unrefinement to the next coarser level.
- The next loop (line 4) is executed while there are eligible cells and its previous iteration succeeded to unrefine at least one cell in $\mathcal{E}^\ell$. In the next line (no. 6), we use the set $\mathcal{E}^\ell$ to initialize an auxiliary set $\hat{\mathcal{E}}^\ell$ of cells.
- The loop, which starts in line 7, visits all the cells that are eligible for the unrefinement. There are various possibilities of choosing the order of visiting the cells. This will be explored in Section 6.2.2.
- A temporary hierarchy is built in line 9. It consists of the temporary subdomains

$$\hat{\Omega}^k = \begin{cases} \overline{\Omega^{\ell+1} \setminus c} & \text{if} \quad k = \ell+1 \\ \Omega^k & \text{otherwise,} \end{cases}$$

  which form the sequence

$$\hat{\mathbf{\Omega}} = (\hat{\Omega}^\ell)_{\ell=1,\dots,N+1}$$

  of nested subdomains, i.e.,

$$[0,1]^2 = \hat{\Omega}^1 \supseteq \hat{\Omega}^2 \supseteq \dots \supseteq \hat{\Omega}^\ell \supseteq \dots \supseteq \hat{\Omega}^N \supseteq \hat{\Omega}^{N+1} = \emptyset \,.$$

  The temporary hierarchy defines the *temporary hierarchical spline space* $\hat{\mathcal{H}} = \operatorname{span} \hat{H}$. Since $\Omega^\ell \supseteq \Omega^{\ell+1} \supseteq \hat{\Omega}^{\ell+1}$, due to the Theorem 1 this leads to a reduced number of degrees of freedom in accordance with Objective (MO).
- In the next step we apply three methods of increasing complexity to construct the temporary parameterization. More precisely, the function **Projection[*i*]** does a local least squares fitting of the geometry on the coarse space $\hat{\mathcal{H}}$ defined by the temporary hierarchy for $i = 1$, and it performs optimization with respect to different classes of functionals for $i = 2$ and $i = 3$, respectively, if the previous iteration did not provide us a temporary parameterization satisfying the conditions. The aim of this approach is to keep the computational costs as low as possible by always using the computationally cheapest projection method. Section 5 describes these stages in detail.
- After this step is executed, line 12 calls the function valid($\hat{\mathbf{p}}, \mathbf{p}_0, \varepsilon$, F) that analyzes whether the temporary parameterization satisfies the objection (MO) and fulfills conditions (C1)-(C3). It returns *true* if conditions (C1)-(C3) hold, and *false* otherwise. This function decides if the temporary parameterization is acceptable.
- If the conditions are fulfilled, then the temporary parameterization $\hat{\mathbf{p}}$, the temporary hierarchy $\hat{\mathbf{\Omega}}$ and the set of the remaining eligible cells $\hat{\mathcal{E}}^\ell$ are accepted. This decision is recorded by assigning the value *true* to the *success* variable. We then proceed to the next cell.
- Otherwise, if the modified parameterization $\hat{\mathbf{p}}$ is still not regular or does not approximate $\mathbf{p}_0$ within the given tolerances, the parameterization $\mathbf{p}$ and the hierarchy Ω remain unchanged and the algorithm also proceeds to another eligible cell.
- The algorithm stops after the set of eligible cells has been exhausted and no further progress with respect to unrefinement is to be expected.

5. The projection step

This section describes how we proceed in line 11 of the Algorithm 1. At the first stage, we execute the least squares projection of the modified parameterization $\hat{\mathbf{p}}$ on the $\hat{\mathcal{H}}$. More precisely, the following functional is minimized locally:

$$\int_{\Delta(c)} \left\| \mathbf{p}_0 - \hat{\mathbf{p}} \right\|_E^2 \,, \tag{5}$$

where $\|.\|_E$ denotes the Euclidean norm. "Locally" means that only control points in a certain vicinity of the unrefined cell are considered by the optimization procedure, hence it suffices to perform integration with respect to a certain subdomain $\Delta(c) \subseteq [0,1]^2$. We choose this (parameterization–dependent) functional since its minimization is computationally simple. Section 5.1 provides a more detailed explanation of this stage.

The second and third stage include the optimization of the certain additional functionals $Q(\mathbf{d})$ (which are called *quality measures*), i.e.,

$$\int_{\Delta(c)} ||\mathbf{p}_0 - \hat{\mathbf{p}}||^2 + Q(\mathbf{d}) \to \min_{\mathbf{d}}, \tag{6}$$

in order to take properties of the parameterization into account. We employ quadratic functionals in the second stage, which is described in Section 5.3, arriving at linear problems (which lead to linear problems). The section provides a more detailed

description. At the third stage, a non–negative linear combination of several non-quadratic quality measures is subject to minimization. The reader is referred to the Section 5.4 for a more detailed explanation of this stage. Further information which is relevant for all the stages, in particular concerning the choice of the local domain $\Delta(c)$, is covered by Section 5.2.

5.1. Local least-squares approximation

This section describes the first stage of the **Projection** function. We use a local spline projector, whose construction is inspired by the recent work of Giust et al. (2020). The minimization of the objective function (5) with respect to a subset of the degrees of freedom (the selection of which is explained in the next section) is used to generate the parameterization

$$\hat{\mathbf{p}}(u,v) = \sum_{i=1}^{N} \sum_{\hat{\tau}^\ell \in \hat{T}^\ell} \hat{\tau}^\ell(u,v)\mathbf{d}_{\hat{\tau}^\ell}$$

that meets the objective (MO) and satisfies conditions (C1)-(C3). Recall that the symbols with the hat refer to the temporary hierarchy.

Based on the chosen domain $\Delta(c)$ (see next section), we split the basis $\hat{T}^\ell$ (for each level ℓ) of the truncated hierarchical spline space into two disjoint sets,

$$\hat{T}^\ell_{\text{int}} = \left\{\hat{\tau}^\ell : \operatorname{supp}\hat{\tau}^\ell \subseteq \Delta(c)\right\} \quad \text{and} \quad \hat{T}^\ell_{\text{ext}} = \left\{\hat{\tau}^\ell : \operatorname{supp}\hat{\tau}^\ell \not\subseteq \Delta(c)\right\},$$

that contain the functions whose support (in the sense of (2)) is contained within the domain $\Delta(c)$ or not, respectively. The control points of the basis functions of the latter set remain unchanged,

$$\hat{\mathbf{d}}_{\hat{\tau}^\ell} = \mathbf{d}_{\hat{\tau}^\ell} \quad \forall \hat{\tau}^\ell \in \hat{T}^\ell_{\text{ext}}\,. \tag{7}$$

We need to find the remaining coefficients $\mathbf{d}_{\hat{\tau}^\ell}$, $\hat{\tau}^\ell \in \hat{T}^\ell_{\text{int}}$.

We consider Gauss points (u_k, v_k) on $\Delta(c)$ and associated weights ω_k for $k \in K$, where K is a finite index set. The over-constrained linear system $A\hat{\mathbf{D}} = \mathbf{F}$ is then solved in the least-squares sense, where

$$A = \left(\sqrt{\omega_k}\hat{\tau}^\ell(u_k,v_k)\right)_{k\in K,\hat{\tau}^\ell\in\hat{T}^\ell_{\text{int}},\ell=1,\dots,N}, \quad \hat{\mathbf{D}} = \left(\hat{\mathbf{d}}_{\hat{\tau}^\ell}\right)_{\hat{\tau}^\ell\in\hat{T}^\ell_{\text{int}},\ell=1,\dots,N},$$

$$\mathbf{F} = \left(\sqrt{\omega_k}\left(\mathbf{f}(u_k,v_k) - \sum_{\ell=1}^{N}\sum_{\hat{\tau}^\ell\in\hat{T}^\ell_{\text{ext}}}\hat{\tau}^\ell(u_k,v_k)\mathbf{d}_{\hat{\tau}^\ell}\right)\right)_{k\in K}.$$

The matrices $A, \hat{\mathbf{D}}$ and $\mathbf{F}$ have the dimension

$$|K| \times \sum_{\ell=1}^{N}|\hat{T}^\ell_{\text{int}}|, \quad \sum_{\ell=1}^{N}|\hat{T}^\ell_{\text{int}}| \times 2 \quad \text{and} \quad |K| \times 2\,,$$

respectively. The resulting function

$$\hat{\mathbf{p}}(u,v) = \underbrace{\sum_{\ell=1}^{N}\sum_{\hat{\tau}^\ell\in\hat{T}^\ell_{\text{int}}}\hat{\tau}^\ell(u,v)\mathbf{d}_{\hat{\tau}^\ell}}_{=\hat{\mathbf{p}}_{\text{int}}(u,v)} + \underbrace{\sum_{\ell=1}^{N}\sum_{\hat{\tau}^\ell\in\hat{T}^\ell_{\text{ext}}}\hat{\tau}^\ell(u,v)\hat{\mathbf{d}}_{\hat{\tau}^\ell}}_{=\hat{\mathbf{p}}_{\text{ext}}(u,v)}, \tag{8}$$

minimizes the objective function (5) among all the functions of the form (8) with control points taking the values (7) in the part $\hat{\mathbf{p}}_{\text{ext}}$, where the integral is evaluated by the chosen Gauss quadrature rule.

5.2. Choice of the subdomain $\Delta(c)$

After selecting the cell c, we need to specify the subdomain $\Delta(c)$ that is considered for the optimization. We construct it as follows. Let the coordinates of the lower left and upper right corners of the cell c be (u_1, v_1) and (u_2, v_2), respectively. We use a *locality parameter* $\lambda \in \mathbb{N} \cup \{0\}$. Then we consider the coordinates

$$\begin{aligned} \hat{u}_1 &= \max(u_1 - \lambda(u_2-u_1), 0), \quad \hat{v}_1 = \max(v_1 - \lambda(v_2-v_1), 0)\,, \\ \hat{u}_2 &= \min(u_2 + \lambda(u_2-u_1), 1), \quad \hat{v}_1 = \min(v_2 + \lambda(v_2-v_1), 1)\,, \end{aligned} \tag{9}$$

and use them to define the *extended cell*

$$E(c) = [\hat{u}_1, \hat{v}_1] \times [\hat{u}_2, \hat{v}_2] \,.$$

The maximums and minimums in (9) ensure that $E(c) \subseteq [0,1]^2$. The set $E(c)$ consists of the cell c and at most λ neighboring cells of the same level in each direction. Then we consider all the THB-splines that are non-zero on E and use the *axis-aligned bounding box* of the union of their supports to define the *restricted domain*:

$$\Delta(c) = \text{aabb}\left(\bigcup_{\text{supp } \hat{\tau}^\ell \cap E(c) \neq \emptyset} \text{supp } \hat{\tau}^\ell \right) . \tag{10}$$

The locality parameter λ controls the size of the domain that used for selecting the THB-splines. Obviously, this domain satisfies $E(c) \subseteq \Delta(c)$. On the one hand, we want to keep the locality parameter λ as small as possible, in order to minimize the computational effort. On the other hand, this parameter should be as large as required to obtain good fitting and optimization results.

If we choose $\lambda = 0$, the set $E(c)$ coincides with the cell c and only THB-splines that are non-zero on the cell selected. The experimental section provides more insights regarding the choice of the locality parameter λ.

5.3. Optimization using quadratic objective functions

This section is devoted to the second stage of the projection function. We use the methods described in (Falini et al., 2015), but adapt them to the domain $\Delta(c)$ and to our specific problem. We start with a parameterization (8), for which the conditions (C1)-(C3) are not satisfied. In order to satisfy them, we try to adjust the coefficients $\mathbf{d} = \{\mathbf{d}_{\hat{\tau}^\ell}, \hat{\tau}^\ell \in \hat{T}^\ell_{\text{int}}\}$ via minimization of the objective function (6). The additional term Q is a linear combination of the length functional

$$Q_\ell(\mathbf{d}) = \int_{\Delta(c)} \left(||\hat{\mathbf{p}}_u||^2 + ||\hat{\mathbf{p}}_v||^2 \right) du dv \,,$$

and the uniformity functional

$$Q_u(\mathbf{d}) = \int_{\Delta(c)} \left(||\hat{\mathbf{p}}_{uu}||^2 + ||\hat{\mathbf{p}}_{uv}||^2 + ||\hat{\mathbf{p}}_{vv}||^2 \right) du dv \,.$$

Here $\hat{\mathbf{p}}_u, \hat{\mathbf{p}}_v, \hat{\mathbf{p}}_{uu}, \hat{\mathbf{p}}_{uv}$ and $\hat{\mathbf{p}}_{vv}$ are partial derivatives. The solution of (6) is again found by solving a system of linear equations $A\hat{\mathbf{D}} = \mathbf{F}$, with the matrices $A, \hat{\mathbf{D}}$ and $\mathbf{F}$ of sizes

$$|K| \times \sum_{\ell=1}^{N} |\hat{T}^\ell_{\text{int}}|, \quad \sum_{\ell=1}^{N} |\hat{T}^\ell_{\text{int}}| \times 2 \quad \text{and} \quad |K| \times 2 \,,$$

respectively. The system which is derived similarly to the previous section. Experimental results that show the performance of this procedure will be reported in Section 6.2.

5.4. Optimization using more general objective functionals

Finally, we describe the third stage of the **Projection** function. We are looking for such coefficients $\mathbf{d} = \{\mathbf{d}_{\hat{\tau}^\ell}, \hat{\tau}^\ell \in \hat{T}^\ell_{\text{int}}\}$ such that the parameterization (8) meets the objective (MO) and satisfies the conditions (C1)-(C3). We are now using additional functionals, which are non-quadratic. There is a wide variety of possibly non-quadratic functions, including

1. the skewness functional

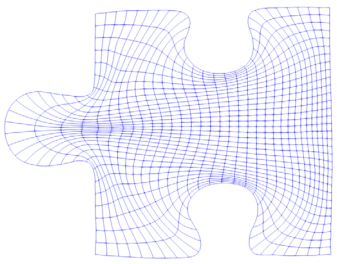
$$Q_s(\mathbf{d}) = \int_{\Delta(c)} \left(\frac{\left(\hat{\mathbf{p}}_u, \hat{\mathbf{p}}_v\right)^2}{\hat{\mathbf{p}}_u \hat{\mathbf{p}}_u \cdot \hat{\mathbf{p}}_v \hat{\mathbf{p}}_v} \right)^2 du dv \,,$$

2. the eccentricity functional

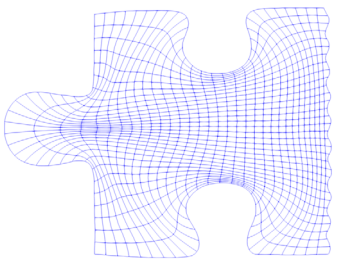
$$Q_e(\mathbf{d}) = \int_{\Delta(c)} \left(\frac{\hat{\mathbf{p}}_u \hat{\mathbf{p}}_{uu}}{\hat{\mathbf{p}}_u \hat{\mathbf{p}}_u} \right)^2 + \left(\frac{\hat{\mathbf{p}}_v \hat{\mathbf{p}}_{vv}}{\hat{\mathbf{p}}_v \hat{\mathbf{p}}_v} \right)^2 du dv \text{ and}$$

3. the area functional

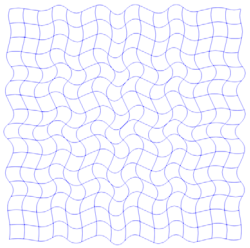
$$Q_a(\mathbf{d}) = \int_{\Delta(c)} \left| \hat{\mathbf{p}}_u \times \hat{\mathbf{p}}_v \right|^2 du dv \,.$$



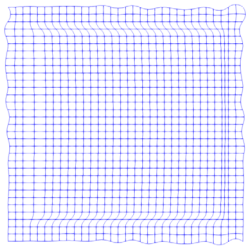
(a) Jigsaw Puzzle, 1156 dofs



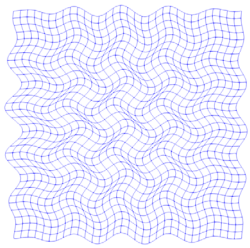
(b) Wiggly Jigsaw Puzzle, 1156 dofs



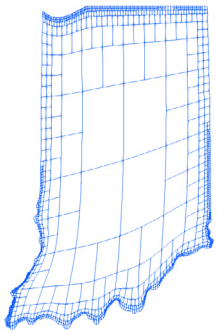
(c) Wiggly Coons Patch, 324 dofs



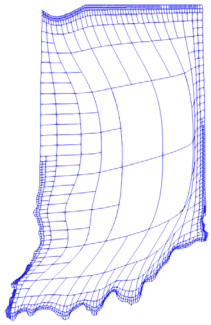
(d) Wiggly Square, 1156 dofs



(e) Wiggly Coons Patch - 3, 1156 dofs



(f) Indiana, 1446 dofs



(g) IndianaIntDist, 1446 dofs

Fig. 1. Initial geometries.

A non-negative linear combination

$$\omega_u Q_u(\mathbf{d}) + \omega_\ell Q_\ell(\mathbf{d}) + \omega_s Q_s(\mathbf{d}) + \omega_e Q_e(\mathbf{d}) + \omega_a Q_a(\mathbf{d}) \to \min_{\mathbf{d}} \tag{11}$$

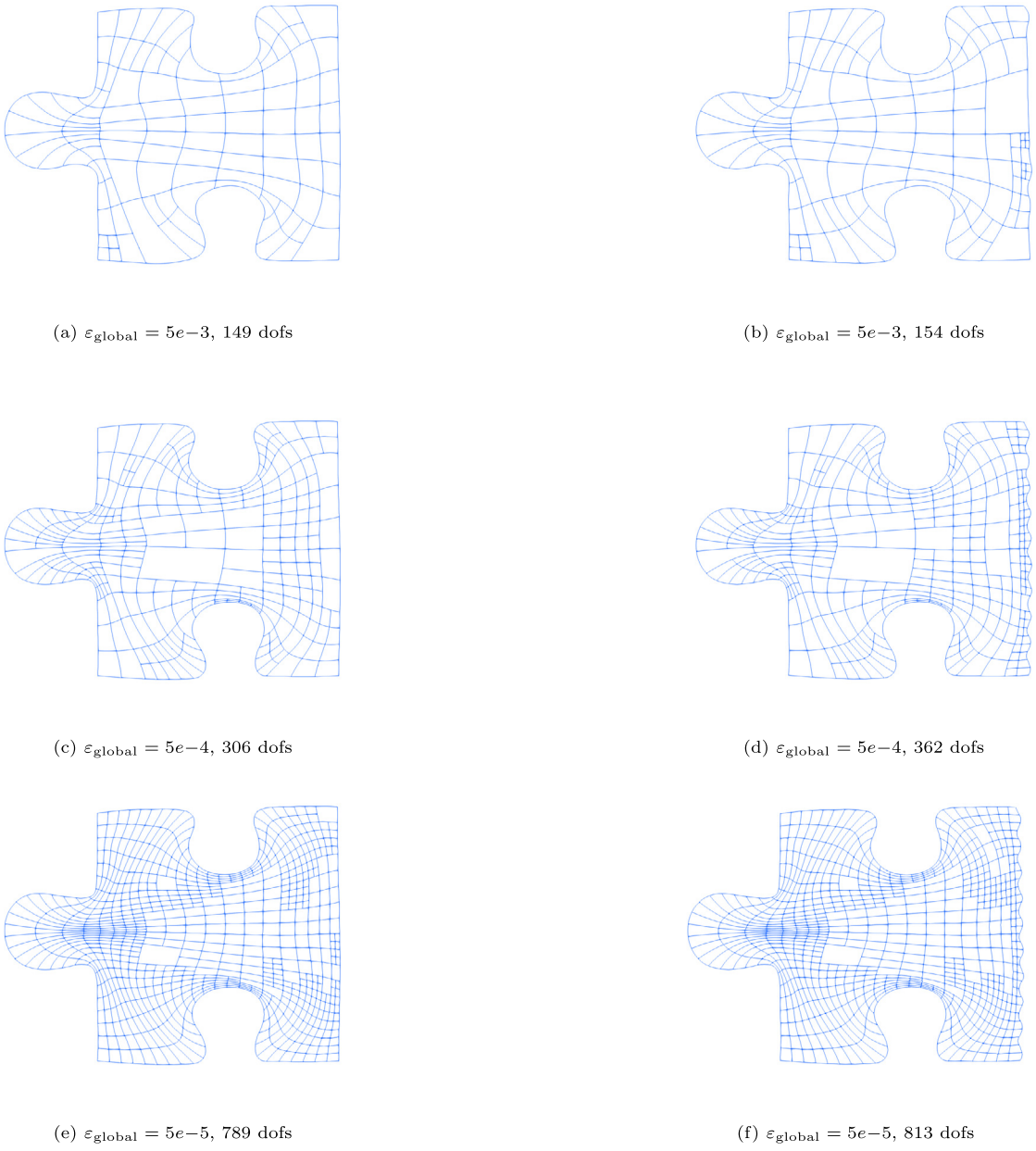
with weights $\omega_u, \omega_\ell, \omega_s, \omega_e, \omega_a$, is used to define the additional term $Q(\mathbf{d})$ in (6). We use a Gauss-Newton-type method in order to find a local minimum of (11). More information of this approach is provided by Falini et al. (2015). In particular, the choice of the weights is described in more detail there. Experimental results will be reported in Section 6.2.

6. Computational results

We use experiments to discuss the effect of the three conditions and of the different parameters of the algorithm. We will concentrate on the conditions (C2) and (C3), which depend on three user-defined quantities, namely $\varepsilon_{\text{global}}, \varepsilon_{\text{feature}}$ and F.

The following geometries, which are depicted in Fig. 1, are used as examples:

1. *Jigsaw Puzzle* (Falini et al., 2015),
2. Jigsaw Puzzle with the oscillation on the eastern boundary (*"Wiggly Jigsaw Puzzle"*),
3. Coons Patches with oscillating boundaries at different levels of detail (*"Wiggly Coons Patch"* and *"Wiggly Coons Patch - 3"*, respectively),
4. a square with wiggly boundaries and distortion on the internal control points (*"Wiggly Square"*),



(a) $\varepsilon_{\text{global}} = 5e{-}3$, 149 dofs

(b) $\varepsilon_{\text{global}} = 5e{-}3$, 154 dofs

(c) $\varepsilon_{\text{global}} = 5e{-}4$, 306 dofs

(d) $\varepsilon_{\text{global}} = 5e{-}4$, 362 dofs

(e) $\varepsilon_{\text{global}} = 5e{-}5$, 789 dofs

(f) $\varepsilon_{\text{global}} = 5e{-}5$, 813 dofs

Fig. 2. First experiment – The effect of the global error. Jigsaw Puzzle is in the left column, Wiggly Jigsaw Puzzle is in the right column. The initial geometries have 1156 degrees of freedom.

5. Indiana (*"Indiana"*) (Falini et al., 2015), and
6. Indiana with internal distortion (*"IndianaIntDist"*).

The section is organized as follows: the first part is devoted to the influence of the quantities that define the three conditions, and the second part analyzes how the results depend on the different parameters of the algorithm.

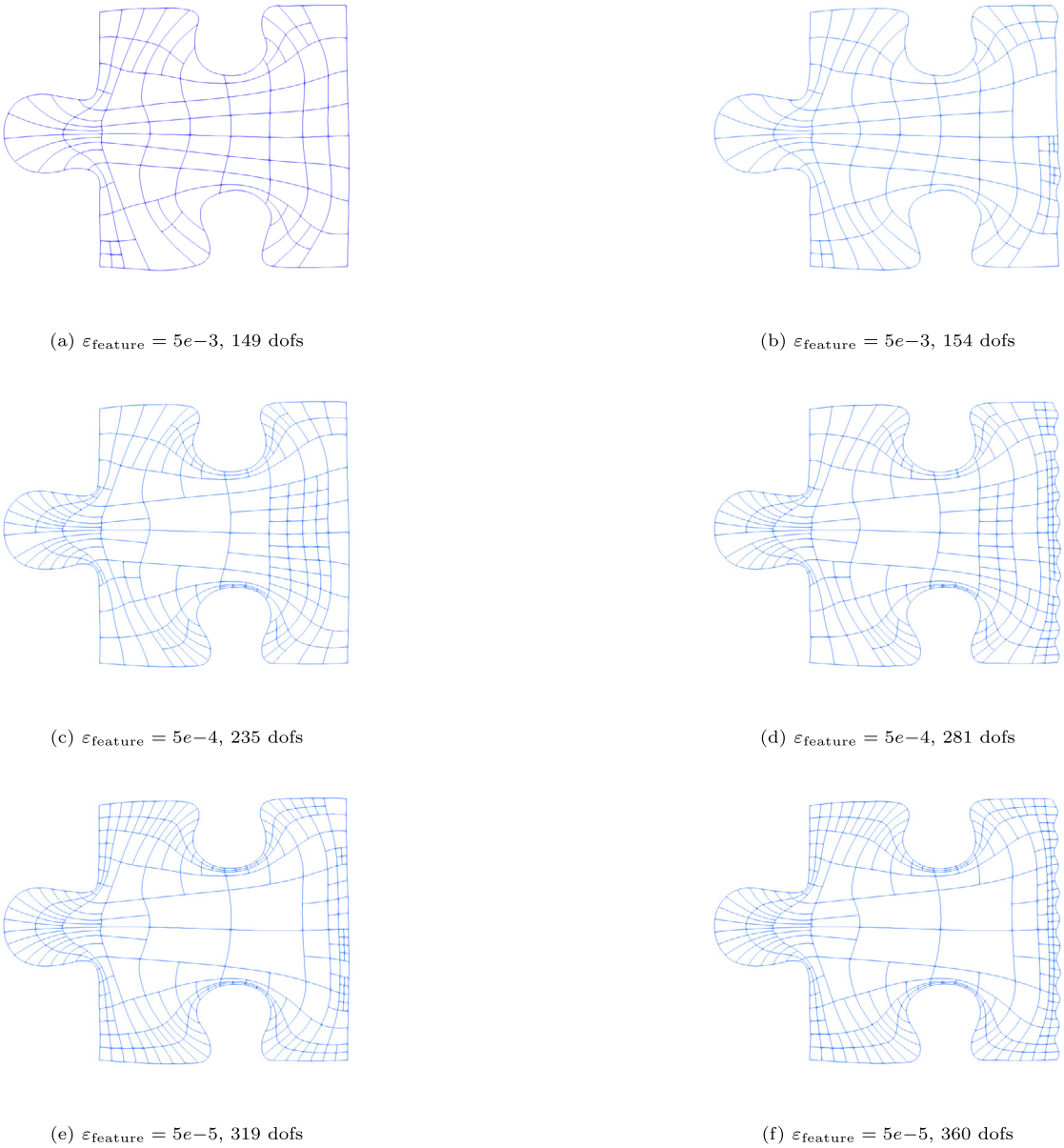
6.1. Conditions (C2) and (C3)

In order to investigate the effect of the conditions (C2) and (C3), we perform three experiments using Jigsaw Puzzle and Wiggly Jigsaw Puzzle. We employ the lexicographic selection of the cells and put $\lambda = 32$.

In the *first experiment*, we set $F = \emptyset$ and do not consider $\varepsilon_{\text{feature}}$. We apply coarsening with different values of $\varepsilon_{\text{global}}$. The results are shown in Fig. 2 and in columns 2 to 4 of Table 1. It can be observed that the lower the global error is, the more degrees of freedom are needed in the resulting geometry, as expected.

Table 1
The number of degrees of freedom for the first, second and third experiment. The initial geometries have 1156 degrees of freedom. The abbreviation u.h. means upper half.

Geometry	First experiment $F = \emptyset$			Second experiment $F = [0,1]^2$, $\varepsilon_{\text{global}} = 1$			Third experiment $F =$ u. h., $\varepsilon_{\text{global}} = 1$		
	$\varepsilon_{\text{global}}$			$\varepsilon_{\text{feature}}$			$\varepsilon_{\text{feature}}$		
	$5e{-}3$	$5e{-}4$	$5e{-}5$	$5e{-}3$	$5e{-}4$	$5e{-}5$	$5e{-}3$	$5e{-}4$	$5e{-}5$
Jigsaw Puzzle	149	306	789	149	235	319	100	129	185
Wiggly Jigsaw Puzzle	154	362	813	154	281	360	111	163	210



(a) $\varepsilon_{\text{feature}} = 5e{-}3$, 149 dofs

(b) $\varepsilon_{\text{feature}} = 5e{-}3$, 154 dofs

(c) $\varepsilon_{\text{feature}} = 5e{-}4$, 235 dofs

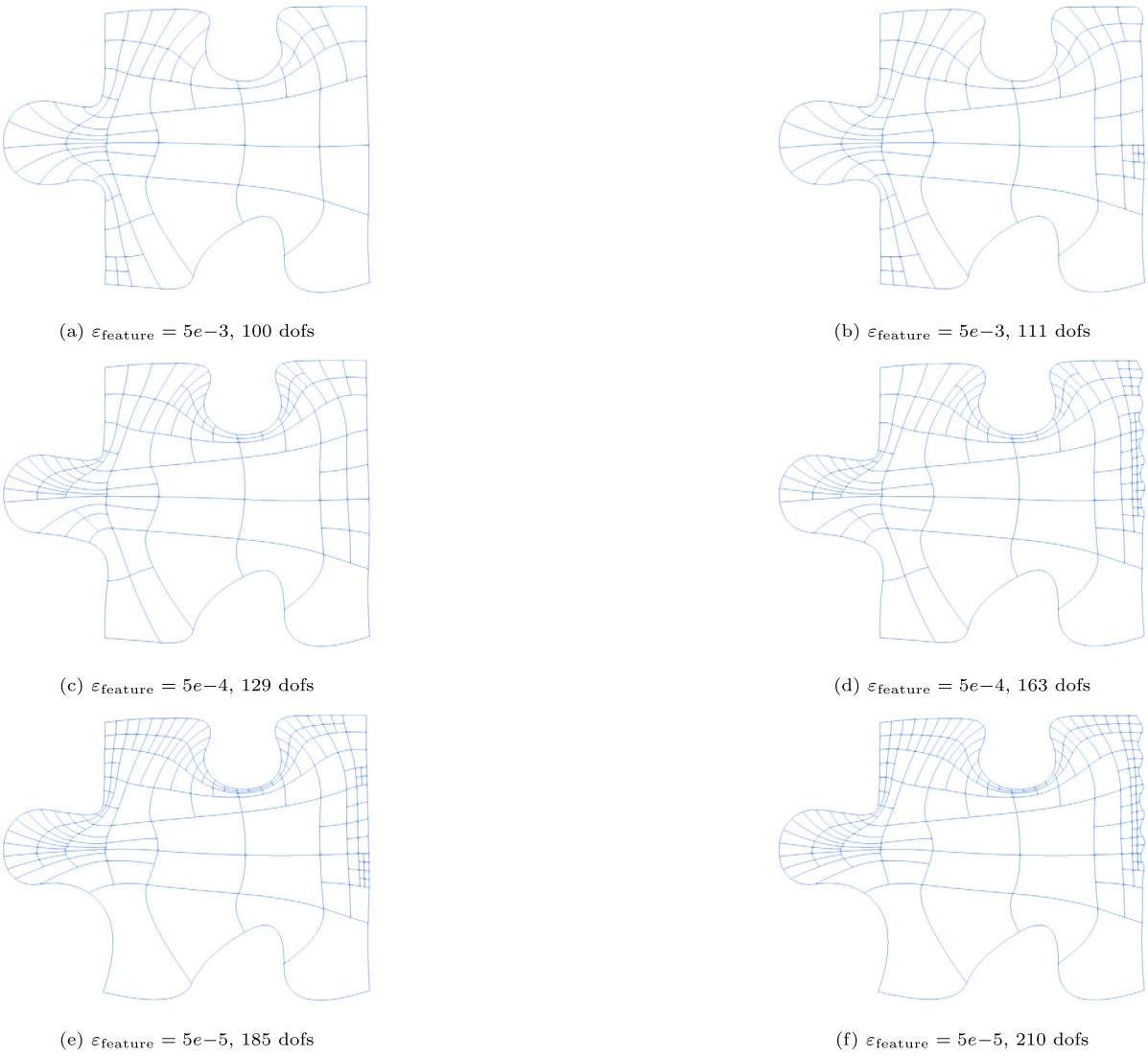
(d) $\varepsilon_{\text{feature}} = 5e{-}4$, 281 dofs

(e) $\varepsilon_{\text{feature}} = 5e{-}5$, 319 dofs

(f) $\varepsilon_{\text{feature}} = 5e{-}5$, 360 dofs

Fig. 3. Second experiment – Error threshold applied to the entire boundary. Jigsaw Puzzle is in the left column, Wiggly Jigsaw Puzzle is in the right column. The initial geometries have 1156 degrees of freedom.

In the *second experiment*, we set $F = \partial[0,1]^2$ and $\varepsilon_{\text{global}} = \infty$ and vary $\varepsilon_{\text{feature}}$. The decrease in the $\varepsilon_{\text{feature}}$ leads to the expected result, as well; we need more degrees of freedom. The effect is demonstrated on Fig. 3 and Table 1, columns 5 to 7.



(a) $\varepsilon_{\text{feature}} = 5e{-}3$, 100 dofs

(b) $\varepsilon_{\text{feature}} = 5e{-}3$, 111 dofs

(c) $\varepsilon_{\text{feature}} = 5e{-}4$, 129 dofs

(d) $\varepsilon_{\text{feature}} = 5e{-}4$, 163 dofs

(e) $\varepsilon_{\text{feature}} = 5e{-}5$, 185 dofs

(f) $\varepsilon_{\text{feature}} = 5e{-}5$, 210 dofs

Fig. 4. Third experiment – Error on the part of the boundary as the featured part, $F = \partial([0,1] \times [0.5,1]) \cap \partial([0,1]^2)$. Jigsaw Puzzle is in the left column, Wiggly Jigsaw Puzzle is in the right column. The initial geometries have 1156 degrees of freedom.

The *third experiment* uses a different feature region F, which is taken here as the upper half of the boundary. The same effect as in the previous two Experiments can be observed if $\varepsilon_{\text{feature}}$ varies: we need more degrees of freedom as the error is decreases, see Fig. 4 and Table 1, columns 8 to 10.

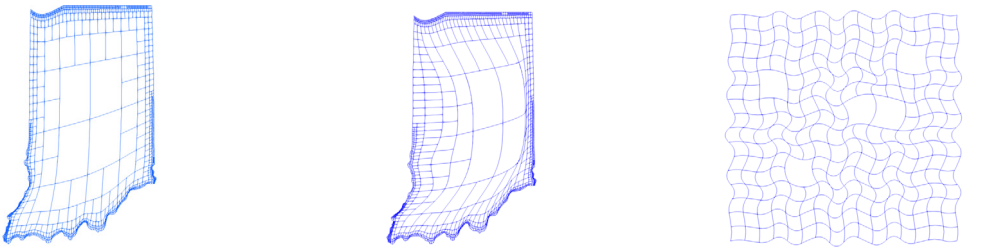
Summing up, these experiments indicate that the stronger the conditions, the more degrees of freedom are needed to satisfy them, see Table 1. While this is not very surprising, it shows that the algorithms work as expected.

6.2. The effect of the different components of the algorithm

This section is devoted to the demonstration of the effect of the different parts of the algorithm. Firstly, we study the effect of employing the different stages of the **Projection** function. Secondly, we show how the method of the choice of the cells affects the result of the coarsening. Finally, we analyze the choice of the parameter λ.

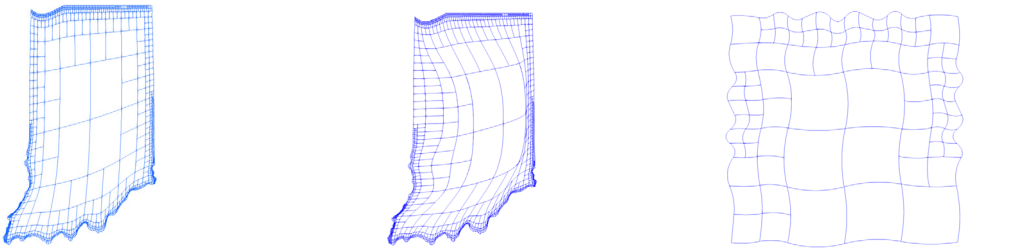
6.2.1. Using the different stages of the Projection function

We compare the different stages of the optimization and least-squares fitting. The following examples are used: Indiana, IndianaIntDist and Wiggly Coons Patch. We select the cells in lexicographic order and use $\lambda = 256$ for Indiana and 16 for Wiggly Coons Patch. The global error $\varepsilon_{\text{global}} = 1$ in all the experiments. The featured part is taken as $\partial([0,1] \times [0,0.25]) \cap \partial([0,1]^2)$ for Indiana and IndianaIntDist and as an upper half of the domain in the case of Wiggly Coons Patch. The error on the featured part was taken as 0.01, 0.004 and 0.01, respectively.



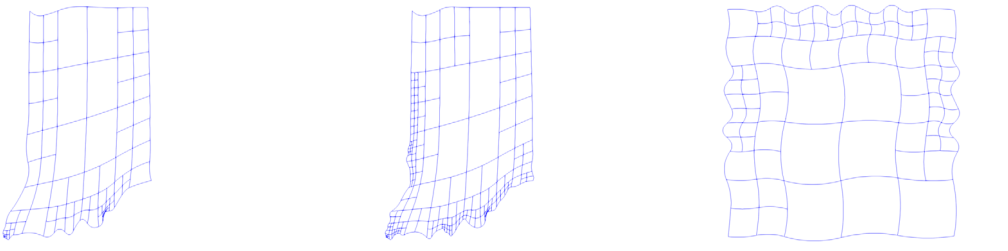
(a) Indiana, 1446 dofs (b) IndianaIntDist, 1446 dofs (c) Wiggly Coons Patch, 288 dofs

Fig. 5. Fourth experiment – Result of the coarsening using only least-squares approximation.



(a) Indiana, 1423 dofs (b) IndianaIntDist, 1423 dofs (c) Wiggly Coons Patch, 100 dofs

Fig. 6. Fifth experiment – Result of the coarsening using linear optimization techniques.



(a) Indiana, 112 dofs (b) IndianaIntDist, 215 dofs (c) Wiggly Coons Patch, 100 dofs

Fig. 7. Sixth experiment – Result of the coarsening using non-linear optimization techniques.

Fig. 5 shows the results of the *fourth experiment*. We perform only least-squares fitting without applying the optimization. As it can be seen, no coarsening is possible for Indiana and IndianaIntDist, and only little coarsening is done for Wiggly Coons Patch.

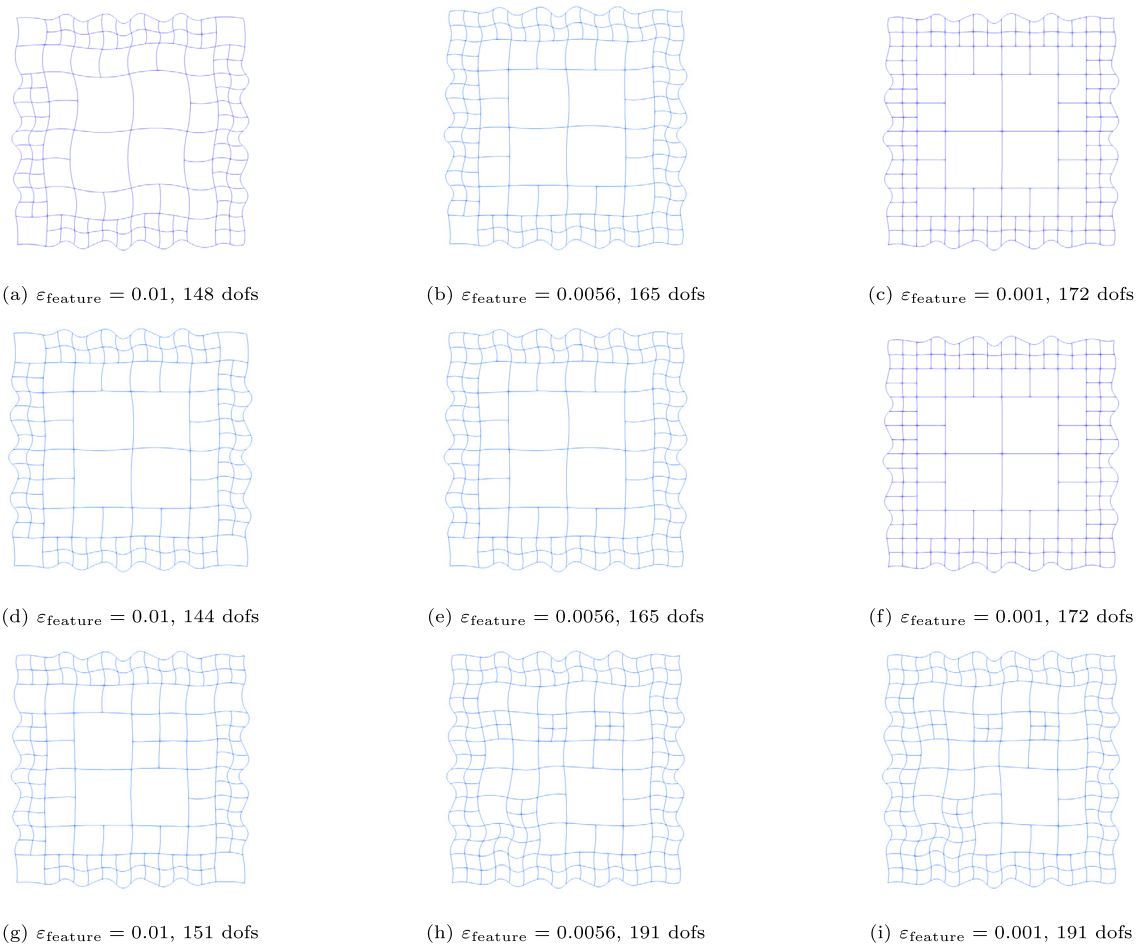
In the *fifth experiment*, we also apply linear optimization. In the case of the Indiana examples, the linear optimization does not help, as shown in Fig. 6. However, it leads to improvements in the case of the Wiggly Coons Patch example. In case of the Coons Patch, we choose $\omega_u = 1$ and $\omega_\ell = 0$. No suitable weights were found for the Indiana examples.

In the *sixth experiment* we perform the optimization with non-linear objective function. The following weights are used: $\omega_s = 1e{+}6, \omega_e = 1e{-}3, \omega_a = 1, \omega_\ell = \omega_u = 0$. The results depicted on Fig. 7 are of particular interest: they show that the linear optimization is sufficient for the case of Wiggly Coons Patch, and non-linear optimization gives no additional improvement in this case. The reason is in the distortion of the geometry caused by the non-linear optimization – since the boundary is more complicated in this case, it is more sensitive to such distortions, as well. However, non-linear optimization allows us to perform significant coarsening in the case of Indiana and IndianaIntDist.

Table 2 summarizes the results. The second column (marked by "none") shows the numbers of degrees of freedom of the initial geometries. The computational time for optimization with quadratic objective functions takes roughly 1.5 more time than least-squares approximation, and the computational time for optimization with non-quadratic objective functions takes roughly 5.25 more time than the optimization with quadratic objective functions. The design of the coarsening procedure ensures that one always uses the fastest available method, in order to speed up the overall computation. As expected, increasing the complexity of the methods makes it possible to achieve additional coarsening.

Table 2
Number of degrees of freedom for the fourth, fifth and sixth experiment.

Geometry	Stage of the **Projection** function			
	none	1	2	3
Indiana	1423	1423	1423	112
IndianaIntDist	1423	1423	1423	215
WigglyCoonsPatch	324	288	100	100



(a) $\varepsilon_{\text{feature}} = 0.01$, 148 dofs (b) $\varepsilon_{\text{feature}} = 0.0056$, 165 dofs (c) $\varepsilon_{\text{feature}} = 0.001$, 172 dofs

(d) $\varepsilon_{\text{feature}} = 0.01$, 144 dofs (e) $\varepsilon_{\text{feature}} = 0.0056$, 165 dofs (f) $\varepsilon_{\text{feature}} = 0.001$, 172 dofs

(g) $\varepsilon_{\text{feature}} = 0.01$, 151 dofs (h) $\varepsilon_{\text{feature}} = 0.0056$, 191 dofs (i) $\varepsilon_{\text{feature}} = 0.001$, 191 dofs

Fig. 8. The seventh (top row), eighth (middle row) and ninth (bottom row) experiment for Wiggly Coons Patch.

6.2.2. Choosing the order of the cells

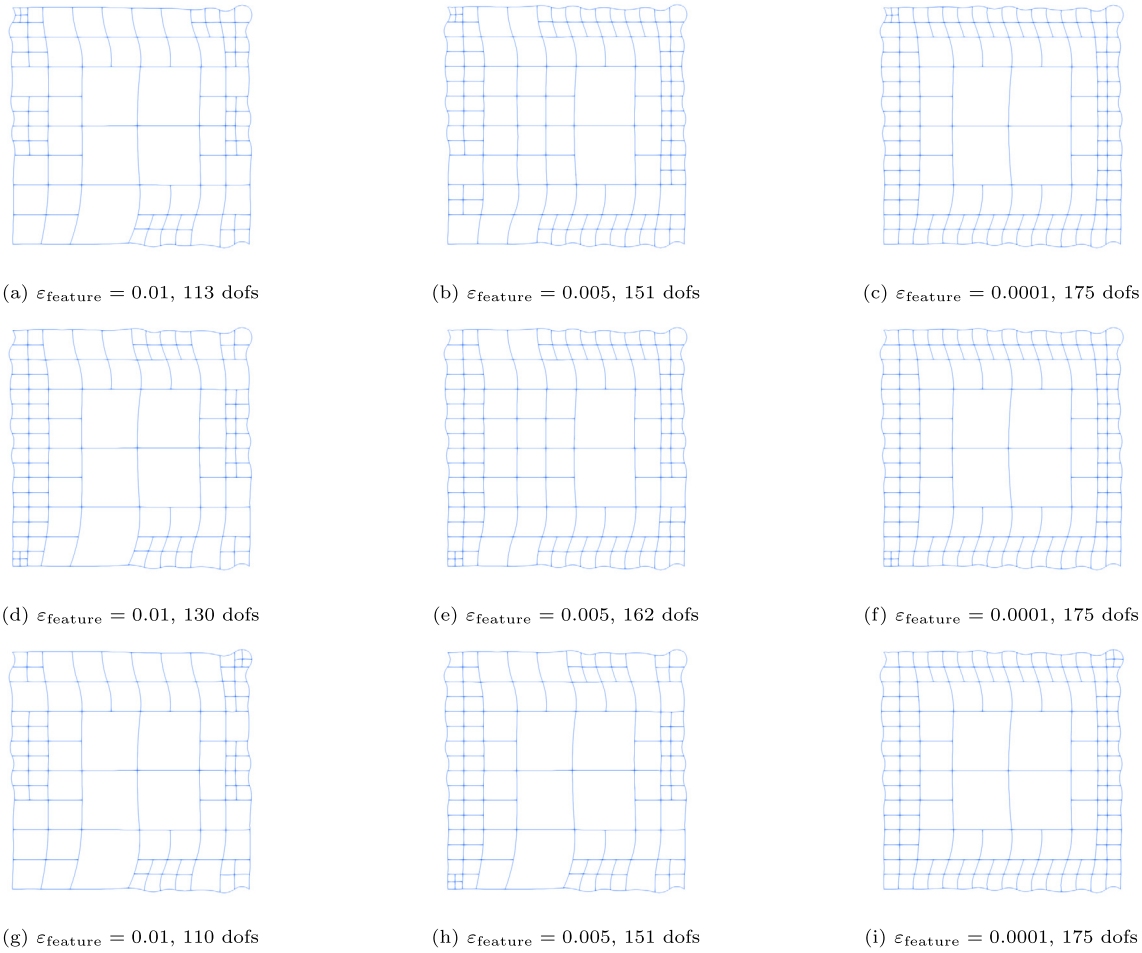
We perform three experiments with different methods of ordering the cells that are eligible for unrefinement (4): we use a randomly generated ordering in the *seventh experiment*, lexicographic ordering in *eighth experiment*, and a semi-random ordering (which will be explained below) in *ninth experiment*. The Wiggly Coons Patch and the Wiggly Square are used as examples. The global error $\varepsilon_{\text{global}} = 1$, F is taken as the upper half of the boundary, and we vary ε_f. The locality parameter λ is taken as 16 for Wiggly Coons Patch and 32 for the Wiggly Square.

The semi-random ordering is based on the outcome of the previous projection step. If it was successful, then we select a next cell from the set (4) which is as close as possible to the current one. Otherwise, we use a random ordering.

The results are shown in Figs. 8 and 9, respectively. The results show that a difference in the algorithms exists, but it shows only in the case of a relatively larger number of cells that are eligible for unrefinement. The results for the Wiggly Coons Patch are nearly identical, especially for small values of $\varepsilon_{\text{feature}}$. Tables 3 and 4 summarize these results.

6.2.3. Selection of the locality parameter

Finally we study the choice of the parameter λ in (9). We use the Wiggly Coons Patch - 3 and Jigsaw Puzzle examples.



(a) $\varepsilon_{\text{feature}} = 0.01$, 113 dofs (b) $\varepsilon_{\text{feature}} = 0.005$, 151 dofs (c) $\varepsilon_{\text{feature}} = 0.0001$, 175 dofs

(d) $\varepsilon_{\text{feature}} = 0.01$, 130 dofs (e) $\varepsilon_{\text{feature}} = 0.005$, 162 dofs (f) $\varepsilon_{\text{feature}} = 0.0001$, 175 dofs

(g) $\varepsilon_{\text{feature}} = 0.01$, 110 dofs (h) $\varepsilon_{\text{feature}} = 0.005$, 151 dofs (i) $\varepsilon_{\text{feature}} = 0.0001$, 175 dofs

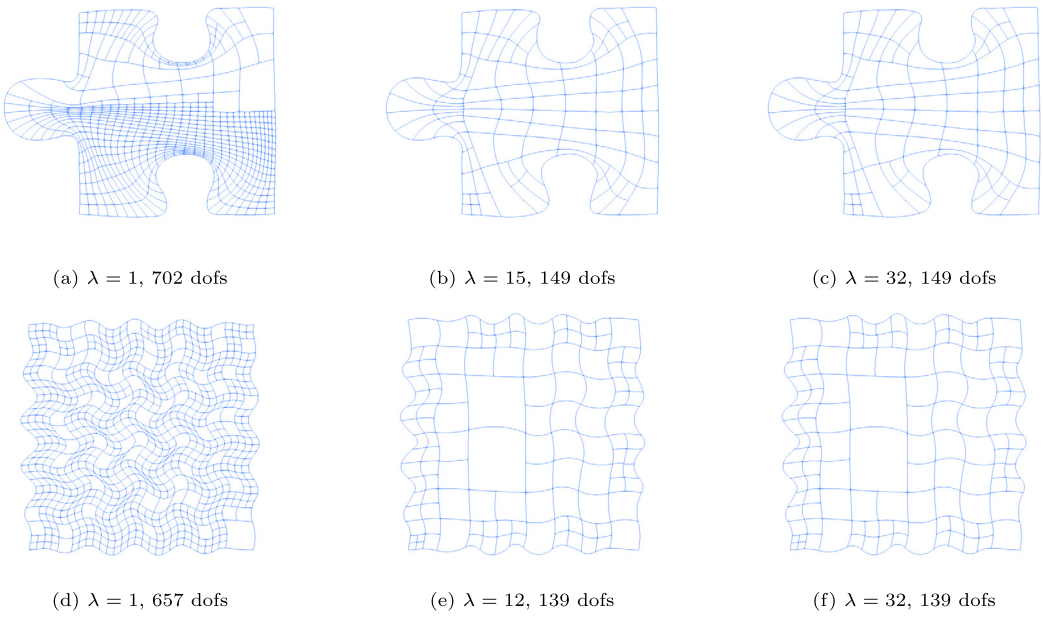
Fig. 9. The seventh (top row), eighth (middle row) and ninth (bottom row) experiment for Wiggly Square.

Table 3
Number of degrees of freedom for seventh, eighth and ninth experiments for Wiggly Coons Patch example; the initial geometry has 324 degrees of freedom.

$\varepsilon_{\text{feature}}$	$F = [0,1]^2$ Cell choice method		
	random	lexicographic	semi-random
0.01	148	144	152
0.0056	165	165	191
0.001	172	172	191

Table 4
Number of degrees of freedom for seventh, eighth and ninth experiment for WigglySquare example; the initial geometry has 1156 degrees of freedom.

$\varepsilon_{\text{feature}}$	$F = [0,1]^2$ Cell choice method		
	random	lexicographic	semi-random
0.01	113	130	110
0.001	151	162	151
0.0001	175	175	175



(a) $\lambda = 1$, 702 dofs (b) $\lambda = 15$, 149 dofs (c) $\lambda = 32$, 149 dofs

(d) $\lambda = 1$, 657 dofs (e) $\lambda = 12$, 139 dofs (f) $\lambda = 32$, 139 dofs

Fig. 10. Tenth and eleventh experiments – Coarsening depending on the parameter λ, $F = \partial([0,1]^2)$, Jigsaw Puzzle in the top row, Wiggly Coons Patch - 3 in the bottom row, the initial geometries have 1156 degrees of freedom.

Table 5
Number of degrees of freedom for Experiments 10 and 11; $\lambda_1 = 1, \lambda_2 = 15$ for Jigsaw Puzzle and 12 for Wiggly Coons Patch - 3, $\lambda_3 = 32$. The initial geometries have 1156 degrees of freedom.

Geometry	$F = [0,1]^2$		
	λ_1	λ_2	λ_3
Jigsaw Puzzle	702	149	149
Wiggly Coons Patch - 3	657	139	139

In *tenth* and *eleventh* experiments, we set $\varepsilon_{\text{feature}} = 5e{-}3$ and $\varepsilon_{\text{feature}} = 1e{-}4$, respectively. We consider the entire boundary and set $\varepsilon_{\text{global}} = 1$. The cells are selected in random order. Fig. 10 and Table 5 show the results. Increasing λ leads to fewer degrees of freedom, but only to some extent (15 and 12, respectively), after which the consequent increase of λ provides no further improvement. At the same time, the computational costs also increase, since the number of degrees of freedom that are considered by the Projection grows with λ. Consequently, there is a trade-off between computational effort and success of the coarsening procedure.

7. Conclusion

We presented an algorithm that creates a coarser geometry from the initial regular geometry so that the coarser geometry approximates the initial geometry with the given level of the precision and is regular. The algorithm is based on the unrefinement of the cells of the parameter domain, on which the geometry is defined, thus obtaining the new subdomain hierarchy.

Future work may include optimum cell selection methods and optimization methods, comparison with other approaches and unrefinement indicators. Instead of the iterative coarsening in our paper, which performs coarsening of one cell at a time, one might try to explore alternative approaches, such as reconstructing the geometry via adaptive refinement according to the given accuracy requirements. Furthermore, the extension to multipatch domains, to surfaces and volumes can also be a topic of further interest, as well as applications to simulation.

CRediT authorship contribution statement

Teymur Heydarov: Software, Validation, Formal analysis, Investigation, Writing
Bert Jüttler: Conceptualization, Writing, Investigation, Supervision, Methodology, Resources
Annalisa Buffa: Conceptualization, Methodology, Funding acquisition

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

Supported by European Research Council through CHANGE project (GA No. 694515).

References

Bracco, C., Giannelli, C., Großmann, D., Sestini, A., 2018. Adaptive fitting with THB-splines: error analysis and industrial applications. Comput. Aided Geom. Des. 62, 239–252.
Buffa, A., Chanon, O., Hernández, R.V., 2020. Analysis-aware defeaturing: first results. Technical Report 11525, arXiv.
Buffa, A., Chanon, O., Vázquez, R., 2022. Analysis-aware defeaturing: problem setting and a posteriori estimation. Math. Models Methods Appl. Sci., 1–44.
Carraturo, M., Giannelli, C., Reali, A., Vázquez, R., 2019. Suitably graded THB-spline refinement and coarsening: towards an adaptive isogeometric analysis of additive manufacturing processes. Comput. Methods Appl. Mech. Eng. 348, 660–679.
Deng, J., Chen, F., Li, X., Hu, C., Tong, W., Yang, Z., Feng, Y., 2008. Polynomial splines over hierarchical T-meshes. Graph. Models 70, 76–86.
Dokken, T., Lyche, T., Pettersen, K.F., 2013. Polynomial splines over locally refined box-partitions. Comput. Aided Geom. Des. 30, 331–356.
Evans, E., Scott, M., Li, X., Thomas, D., 2015. Hierarchical T-splines: analysis-suitability, Bézier extraction, and application as an adaptive basis for isogeometric analysis. Comput. Methods Appl. Mech. Eng. 284, 1–20.
Falini, A., Špeh, J., Jüttler, B., 2015. Planar domain parameterization with THB-splines. Comput. Aided Geom. Des. 35, 95–108.
Forsey, D.R., Bartels, R.H., 1988. Hierarchical B-spline refinement. In: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, pp. 205–212.
Garau, E.M., Vázquez, R., 2018. Algorithms for the implementation of adaptive isogeometric methods using hierarchical B-splines. Appl. Numer. Math. 123, 58–87.
Giannelli, C., Jüttler, B., Speleers, H., 2012. THB-splines: the truncated basis for hierarchical splines. Comput. Aided Geom. Des. 29, 485–498.
Giust, A., Jüttler, B., Mantzaflaris, A., 2020. Local (T) HB-spline projectors via restricted hierarchical spline fitting. Comput. Aided Geom. Des. 80, 101865.
Hennig, P., Ambati, M., De Lorenzis, L., Kästner, M., 2018. Projection and transfer operators in adaptive isogeometric analysis with hierarchical B-splines. Comput. Methods Appl. Mech. Eng. 334, 313–336.
Hughes, T.J., 2008. Isogeometric analysis: Progress and challenges. ABTEKNILLINEN KORKEAKOULU, 22.
Hughes, T.J., Cottrell, J.A., Bazilevs, Y., 2005. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput. Methods Appl. Mech. Eng. 194, 4135–4195.
Kiss, G., Giannelli, C., Zore, U., Jüttler, B., Großmann, D., Barner, J., 2014. Adaptive CAD model (re-) construction with THB-splines. Graph. Models 76, 273–288.
Lorenzo, G., Scott, M., Tew, K., Hughes, T., Gomez, H., 2017. Hierarchically refined and coarsened splines for moving interface problems, with particular application to phase-field models of prostate tumor growth. Comput. Methods Appl. Mech. Eng. 319, 515–548.
Scott, M.A., Thomas, D.C., Evans, E.J., 2014. Isogeometric spline forests. Comput. Methods Appl. Mech. Eng. 269, 222–264.
Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A., 2003. T-splines and T-NURCCs. ACM Trans. Graph. 22, 477–484.
Beirão da Veiga, L., Buffa, A., Sangalli, G., Vázquez, R., 2014. Mathematical analysis of variational isogeometric methods. Acta Numer. 23, 157–287.
Vuong, A.V., Giannelli, C., Jüttler, B., Simeon, B., 2011. A hierarchical approach to adaptive local refinement in isogeometric analysis. Comput. Methods Appl. Mech. Eng. 200, 3554–3567.
Xie, X., Yang, A., Jiang, N., Zhao, W., Liang, Z., Wang, S., 2021. Adaptive topology optimization under suitably graded THB-spline refinement and coarsening. Int. J. Numer. Methods Eng. 122, 5971–5998.